Monitoring of a Virtual Infrastructure Testbed

Edgar Magana
CISCO
10 West Tasman Dr.
San Jose, CA, USA
eperdomo@cisco.com

Antonio Astorga, Joan Serrat, Rafael Valle
Universitat Politècnica de Catalunya
Jordi Girona 1-3
Barcelona, Spain
{aastorga, serrat, valle}@tsc.upc.edu

***Abstract*—This paper presents a SNMP-based Monitoring Agents for Multi-Constrain Resource Scheduling in Grids (SBLOMARS) as an effective solution for resource usage monitoring in virtual network environments. SBLOMARS is different to current large-scale distributed monitoring systems in three essential aspects: Firstly, it reaches a high level of generality by the integration of the SNMP protocol and thus, facilitates to handle heterogeneous operating platforms. Secondly, it is able to self-configure the polling periods of the resources to be monitored depending of network context and finally, it makes use of dynamic software structures to interface with third parties, allowing to be deployed in a wide range of devices, from simple mobile access devices to robust multiprocessor systems or clusters with even multiple hard disks and storage partitions. SBLOMARS has been deployed in EmanicsLab, a virtual laboratory constituted by fourteen nodes distributed in seven European Universities. Although the research is not yet concluded, available results confirm its suitability to deal with the challenges of monitoring virtual networks.**

Keywords-Large-scale Distributed Network Monitoring, Virtual Networks, Self-configurable systems

I. INTRODUCTION

Virtualization is called to be a key enabler of the future Internet. The complexity of future Internet services will require the coexistence of different network protocols and network architectures for better end-to-end services and easy management. Therefore the network infrastructure will need to be partitioned to work with different protocols and management strategies. The technology that allows this co-ordinated network splitting is virtualization. Network devices, serves or any other network element will be able to support different and independent virtual devices. Each of these virtual devices will be remotely connected to their peers constituting a virtual network. Therefore, the Internet will be a set of dynamically evolving virtual networks. Setup and tear down of these networks will be on demand of the different communities and actors.

Although there is still a long way to go until this is made a reality, there are already today virtualization mechanisms and virtualization tools especially in the field of workstations and computing devices that can trail research and extend the concept to other network elements like routers and switches. Among the different implementations of virtual concepts existing today one of the most relevant is PlanetLab [1]. Established in 2002 by UC Berkeley, Princeton University, and University of Washington, it consist of hundreds of computers distributed worldwide offering the possibility to its participants to design, deploy and test large scale applications and services. Thus it makes reality the concept of distributed virtualization [2]. Currently it consist of about a thousand of nodes distributed in 35 countries and supporting 500 distinct services, each working in this global lab as it were alone [4]. Nevertheless, the existence of this infrastructure does not mean that the problem of network virtualization is already solved. Virtualization is only provided at the computation level but not at the networking level.

In addition to the challenges imposed by network virtualization itself it is not less true that the orchestration and control of these conceptually independent entities requires the confluence of efficient and scalable monitoring, scheduling and management approaches. Accounting and auditing of resource usage are fundamental in virtual environments [2, 3].

In this research we focus our attention in the monitoring problem of virtual environments, describing the approach and results of a monitoring system deployed in a virtual laboratory based on PlanetLab. Our intention is to design and test a monitoring system able to work in virtual environments. To do so, we use PlanetLab technology but not in the true Planetlab environment but in a reduced scale, fully owned virtual environment. The reason is twofold; first, owing the full infrastructure we are able to tailor the control and virtualization mechanisms to suit much better our own purposes; and on the other hand, this is possible thanks to the existence of the tool that allows for virtualization of resources in the same way as Planetlab does. In other words, this tool allows for the recreation of the PlanetLab environment at a reduced scale and under the full control of its owners with all the corresponding advantages. More specifically, the deployment of the PlanetLab technology was done in the scope of the EMANICS Network of Excellence, an ongoing EU co-funded integration and dissemination project in the field of Network and Services Management[1]. This ad-hoc PlanetLab implementation is called EmanicsLab and will be described in the following sections. The monitoring system we propose is fully distributed, capable to deal with heterogeneous nodes and with self adaptation capabilities to keep track of the relevant events in the network and at the same diminish the monitoring process overhead. The system is already deployed in EmanicLab and therefore we are able to show different aspects of its operation. The paper is structured as follows. After this introduction,

978-1-4244-4388-8/09/$25.00 ©2009 IEEE

Section II presents the main concepts and relevant data behind PlanetLab in our EmanicsLab implementation. Section III is devoted to summarize our monitoring approach. Section IV presents the process of deployment of our application, whereas Section V shows the results obtained and their interpretation. Finally the paper concludes with a summary of our findings and highlighting our plans for the immediate future.

II. EMANICS LAB: A PLANET-BASED VIRTUAL LABORATORY

The basic concept in PlanetLab is that participating nodes are structured into isolated virtual machines (VMs) named *slivers*. Each VM is enabled to use the totality of the available resources in the node or a maximum is fixed by means of a quota. Anyway, a user service or application making use of a sliver in a node will not be aware of applications running into other slivers of the same node; its assigned sliver will be like a dedicated machine. The isolation of resources in VMs is done by means of the Virtual Machine Monitor (VMM), a component of the environment that needs to be installed in the node participating in PlanetLab [2]. The VMM is based in Linux with *vserver* extensions although eventually it could also work with XEN. This environment also includes the local Node Manager (NM), which as its name suggests, it is entrusted of the lifecycle of VMs in the node and an Administration interface [3]. Figure 1 sketches the above described PlanetLab architecture at node level.

User services can require juts one sliver in one node or most likely many slivers in different nodes. The set of VMs assigned to one application is called slice. The slice guarantees the necessary distributed resources to allow the execution of a given service and therefore it can be seen as an overlay on top of the physical network. The real network is then shared by as many slices as user applications are needed.

Management of Planetlab is centralized. The management tool is called PlanetLab Centre (PLC) that is physically run by system administrators at the Princeton University. PLC acts as a trusted intermediary between participating node owners and resource users. The essential activities of the PLC are the access control of users and the allocation of resources to the slices. In addition, the PLC also provides several administration tools that can provide valuable information to users. Among these tools we can mention SliceStat and CoStat [3], which consist of low level programs gathering resource usage information at slice level, and PlanetFlow that consists of an auditing tool providing packet flow information andmapping the network activity to the slices that cause such activity. For a comprehensive and detailed description of different administrative tools the reader is invited to visit [3, 4]

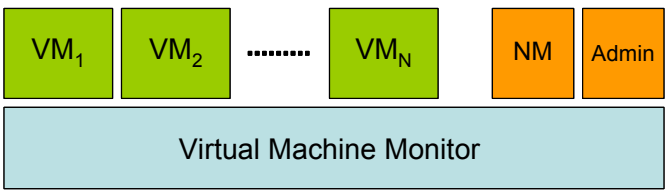

Figure 1. A PlanetLab Node Architecture

Nevertheless, not all are advantages when using PlanetLab. As said above, the management of Planetlab is centralized and based on the trust established by the PLC. This might create a problem when special trust relationships must be set or when strict confidentiality is a must. This has occurred for instance in the EMANICS project [1] when distributing traffic traces generated by third parties through project participants. Thus, when special user needs arise it would be better to have a privately managed network with much better control of user access rights and so on. Another problem of using the public PlanetLab infrastructure is the need to accept restrictions in terms of virtualization platform tools, management service tools and resource quota allocation. In case of the former, there is no clue that current virtualization platform is the best to fit specific service requirements; for instance, a user application would run better if XEN were used instead of the current Linux extensions; but there is no option to replace it. In the second case, imagine for instance that we were wishing to introduce SNMP [6] as a basic building block in a monitoring process at VM or slice level; this would not be possible unless we were able to modify the VMM or the NM and this is not possible when working with the public PLC. Finally, the third case is also relevant when user applications require more resources than the amount allowed by system quotas.

For all the above reasons, it is advisable in some occasions to build a privately owned virtual infrastructure. Fortunately, PlanetLab also allows this because it offers the full fledged PLC as a package that can be installed to recreate the Planetlab platform but deployed only in selected nodes that become a full featured private PlanetLab. In EMANICS, the need to do collaborative research within its fourteen institutions highlighted the opportunity to build a virtual laboratory. In addition, as the specificity above mentioned to build a privately owned virtual lab applied, it was decided to create a private laboratory making use of the PlanetLab deployment tools, calling it EmanicsLab.

EmanicsLab consists today of 14 nodes distributed in 7 sites throughout Europe and supporting about 10 active slices. Figure 2 presents the nodes distribution.

The slices in EmanicsLab serve different research purposes ranging from VoIP applications to traffic traces storage, indexing and search of documents, monitoring and others. Nevertheless, we will focus on the monitoring applications, which is the scope of this paper.

EmanicsLab supports two monitoring slices, one to support Ganglia [7] and another to support SBLOMARS [8]. Ganglia is "a scalable distributed monitoring system for high-performance computing systems such as clusters and Grids. It is based on a hierarchical design targeted at federations of clusters. It leverages widely used technologies such as XML for data representation, XDR for compact, portable data transport, and RRDtool for data storage and visualization. It uses carefully engineered data structures and algorithms to achieve very low per-node overheads and high concurrency" [9]. On the other hand, SBLOMARS (SNMP-based Monitoring Agents for Multi-Constrain Resource Scheduling in Grids) is a pure decentralized monitoring system in charge of permanently capturing computational resource performance based on autonomous distributed agents for multi-constrain resource scheduling in large-scale distributed networks such as Grid.

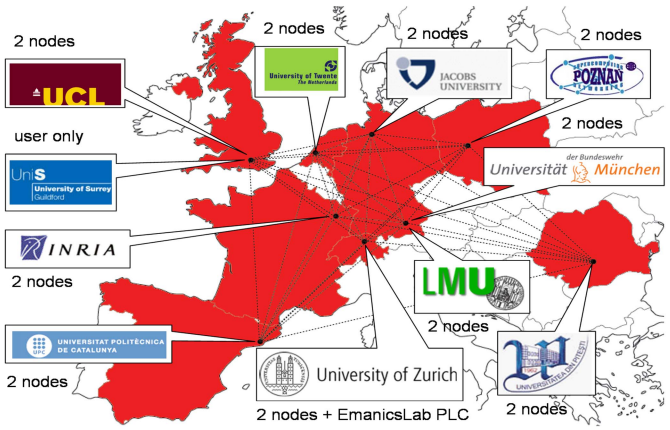

Figure 2. EmanicsLab Nodes Distrubution

III. SBLOMARS

The deployment of the SBLOMARS Monitoring System is done in every participating node. The code is broadcasted along the target network. Once the nodes have received the code, they start the SBLOMARS monitoring agents' instantiation through scripts written into the code itself. This activity is very similar to what occurs when a software disk is inserted into the CD-ROM driver.

A. *Architectural Aspects*

Figure 3 shows the main components and interfaces of SBLOMARS. The Principal Agent Deployer(1) is aimed to deploy a specific monitoring agent for each kind of resource to be monitored. In addition, it offers a generic user interface that can be used to specify thresholds that will determine the time interval between consecutive requests of the SNMP-MIB variables, as well as the number of data values required to calculate a given statistics.

The Resource Sub-Monitoring Agents(2) is the generic agent properly said that is going to be instantiated in as many classes as different kind of resources to be tracked. Currently we have up to six different monitoring agent instantiations: memory, processor, software, storage, network interfaces (at resourcelevel), and end-to-end network connectivity (at network level). The fact that only the necessary agents are instantiated is a key to reduce the global network monitoring activity and consequent overhead.

The Resource Discovery(3) registers the types of resources that are available in the hosting node, storing them in the Network-Map Database (8). The content of this database is broadcasted to be known by other applications (resource schedulers for instance). The Real-Time Report(4) generates real-time resource availability information for each type of resource. This information is published by means of what is denominated Dynamic Software Structures(7). Accessibility to these data structures is granted through network socket connections.

The Historical Report(5) generates statistical resource availability information for each type of resource. This information is published by means of data files called XML Monitoring Service(6). Also the XML reports are accessible through network sockets.

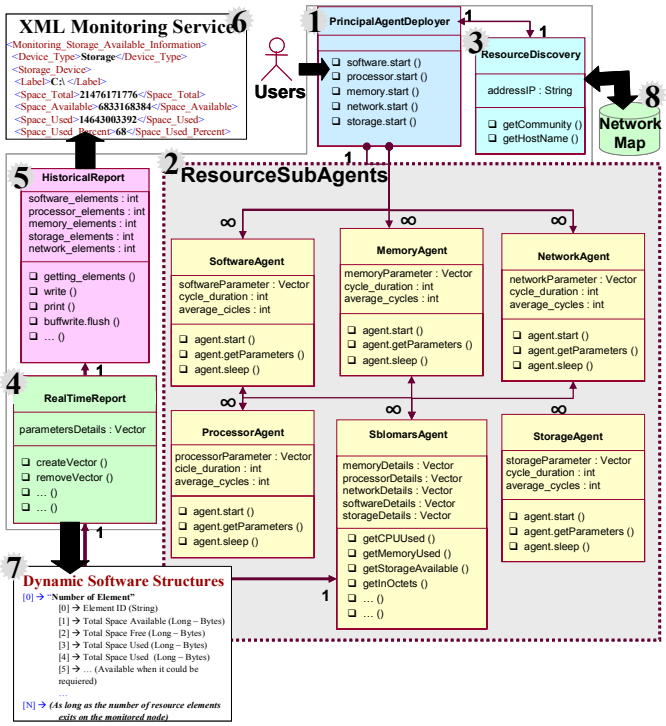

Figure 3. SBLOMARS components and external interfaces.

B. *Distinguishing features*

1) *SNMP-based interaction with devices*

SBLOMARS makes use of the SNMP to get the necessary information from its hosting nodes. Our monitoring agents will retrieve the required information by contacting specific objects of the available MIBs. In particular we have used HOST-RESOURCES-MIB [6], UC-DAVIS-MIB [12], INFORMANT -MIB [13] and CISCO-RTTMON-MIB [10], which are standard and well-known MIBs for networking and computing resources.

2) *Adaptability of polling periods*

SBLOMARS re-configures its polling periods automatically, according to the usage of a given resource. Distributed monitoring systems not exhibiting this property may cause node resource overload (and eventually network traffic overload) when the polling period is excessively short or have an unacceptable probability of state change misdetection when the polling period is made too long.

In our design we have considered this issue. This means that when a resource is being used quite frequently, the time between consecutive requests will be decreased and the amount of monitored information will be much greater. But if a resource is not being used for a long period of time, the time between consecutive readings will be increased and the amount of monitored information will be less. Consequently, network overload will be accommodated accordingly.

The advantages of using this self-adjustment algorithm is not at the expense of a noticeable processor overload penalty [8].

3) *Collection of information from the SBLOMARS agents*

The resource availability information offered by SBLOMARS could be collected by any external entity such as resource scheduler or graphical interfaces. The communication process between the collecting entity and SBLOMARS is done by means of network socket connections. This socket interface is redundantly implemented by each agent running on the hosting node. Thereby, we would have as many socket interfaces as resources available on the hosting node.

IV. THE SCALABILITY ISSUE

One of the main issues in any monitoring system is to verify how scalable the system is. In order to show the scalability of SBLOMARS we have run two experiments. In the first one we have measured the time needed to execute the configuration and activation phases of the SBLOMARS monitoring system along the fourteen nodes of EmanicsLab (Figure 4). In the second experiment we have measured the time needed to collect the availability of one resource per node (CPU) from the entire EmanicLab (Figure 5).

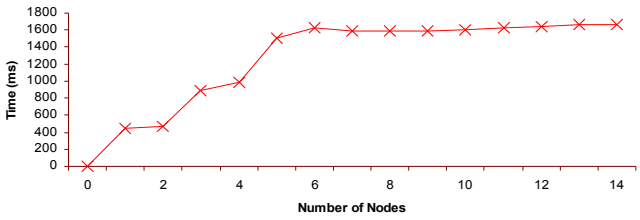

Figure 4. SBLOMARS configuration and activation time vs the number of nodes considered in the EmanicsLab

In both experiments we have repeated fourteen times the corresponding processes, incrementing in steps of one the number of affected nodes.

The resulting graph shows that SBLOMARS increases the configuration time up to six nodes and then it remains independent on the number of nodes. This behaviour is attributed to the impact of the network traffic, especially noticeable when there are only a few nodes. In practice we conclude that the impact of the network size in the system deployment is not significant.

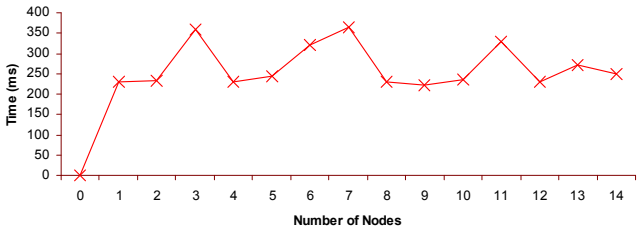

Figure 5. SBLOMARS Collect time vs the number of nodes in the EmanicsLab

In Figure 5 we collect the CPU availability of each node from a central site. This oscillating behaviour is attributed to the fact that the EmanicsLab test-bed has a high level of networking activity that we could not remove for our experiments. The conclusion is that the trend of the average remains constant regardless of the number of nodes in the experiment.

V. RESOURCE AVAILABILITY RESULTS

As already said, SBLOMARS offers real-time and statistical resource availability information by means of JAVA-based sockets connections and XML-based files. This information could appear quite crude or unfriendly for customers, network administrator or resources owners. Therefore, we have integrated a graphical interface to bring user friendly information regarding resource availability to any third party on the network. This graphical interface does not impact the performance of the SBLOMARS agents due to the fact that it is collecting the already available information from databases in every node [8].

The GUI can switch the interval of offered data between different time windows. Figure 6 shows the last hour of activity but we could show the last twenty-four hours of collected information or the full amount of information since SBLOMARS agents were activated.

Unlike the Grid, the use of resources in EmancsLab is available for all partners with no priorities nor scheduling. Therefore, the data offered by SBLOMARS is important to help the virtual network administrator to distribute the use of resources in a load-balanced way. With this application in mind, we did two experiments in order to show that SBLOMARS can be useful to distribute resources load along all the participants in the virtual environment.

The first experiment is intended to show the resource utilization behavior of EmanicsLab with no scheduling constraints. For this experiment we have executed processes through all nodes of the virtual network that are randomly allocated. In Figure 6 we are illustrating the resulting graphs from six nodes. CPU in some nodes is idle whereas in others is fully used; there is an clear unbalance in the network resources usage.

The second experiment is related to show the performance of EmanicsLab nodes but making use of the resource availability information interfaces to pre-select the destination nodes where processes will be dispatched instead of just doing a random selection. Figure 7 shows the results of this experiment. Clearly the processes are making a much more balanced use of the available resources; instead of having the CPUs at 100% or completely idle, the graphs reveal a 40% average utilization.

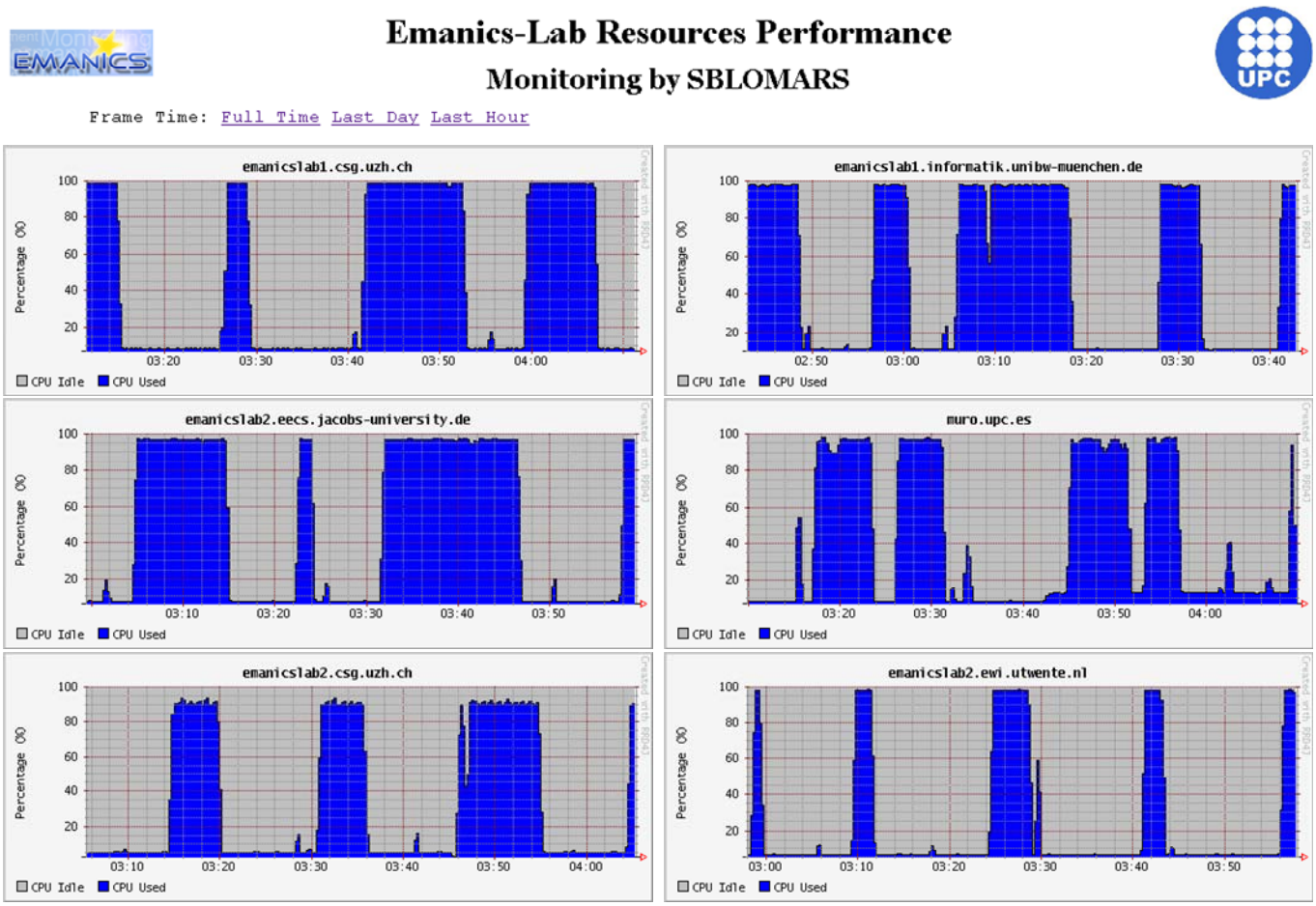

Figure 6. SBLOMARS Resource utilization behavior of six EmanicsLab Nodes with no scheduling constraints.

This behavior is very advisable in virtual environments. In fact, whereas in the first experiment when CPU is at 100% of its capacity the new incoming process will be queued until the CPU has completed previous jobs, in the second experiment it is not necessary to queue processes because nodes have idle CPU and then they have availability to execute more processes.

The penalty to pay in a resource scheduled approach, as the case shown in Figure 7, is the time elapsed in analyzing which nodes are idle and which ones busy; this is also known as the scheduling time. The basic objective of any scheduler is to obtain lower scheduling times than the waiting times when a process is queued. SBLOMARS also allows to measure the scheduling times versus the non-scheduled approach. In subsequent experiments we have measured the elapsed time of a random distribution of the process versus the elapsed time when SBLOMARS is requested in advance to know resource status. It is worth to mention that the criteria to decide which nodes will be selected or not is based on a threshold of CPU utilization of 70%. Results have showed that unconstrained mechanism is less time consuming because there is not previous interaction with the monitoring system. On the contrary, the mechanism that first check the resource status and then sends the process (based on a given allocation process) is clearly more time consuming.

VI. CONCLUSIONS

This paper has shown that SBLOMARS is a solution for monitoring virtual network environments with properties that make it very attractive in front of other large-scale monitoring approaches, namely its flexibility to be practically installed in any operating system platform, its capability of self-adjust the polling period of the resources being monitored and its facilities for exporting monitored resources to data collection third parties. Thanks to this properties, SBLOMARS can be deployed any type of network node, from simple mobile access devices to complex networking and computing systems.

Initially conceived for Grids, SBLOMARS has been adapted in virtual environments making use of EmanicsLab, a virtual laboratory supported by European Universities. The obtained results have revealed that the configuration and activation of SBLOMARS is practically independent of the size of the network. In addition, the time needed to perform a full information collection cycle is also not impacted by the number of network nodes, thanks to the mechanism implemented for data collection. The above results are an indicative of the good scalability properties of our approach. Finally, when SBLOMARS cooperates with a scheduler, the behaviour of both systems show that the network load is well balanced throughout the network, this is a proof that both, the real time and the statistical data obtained by SBLOMARS is reliable.

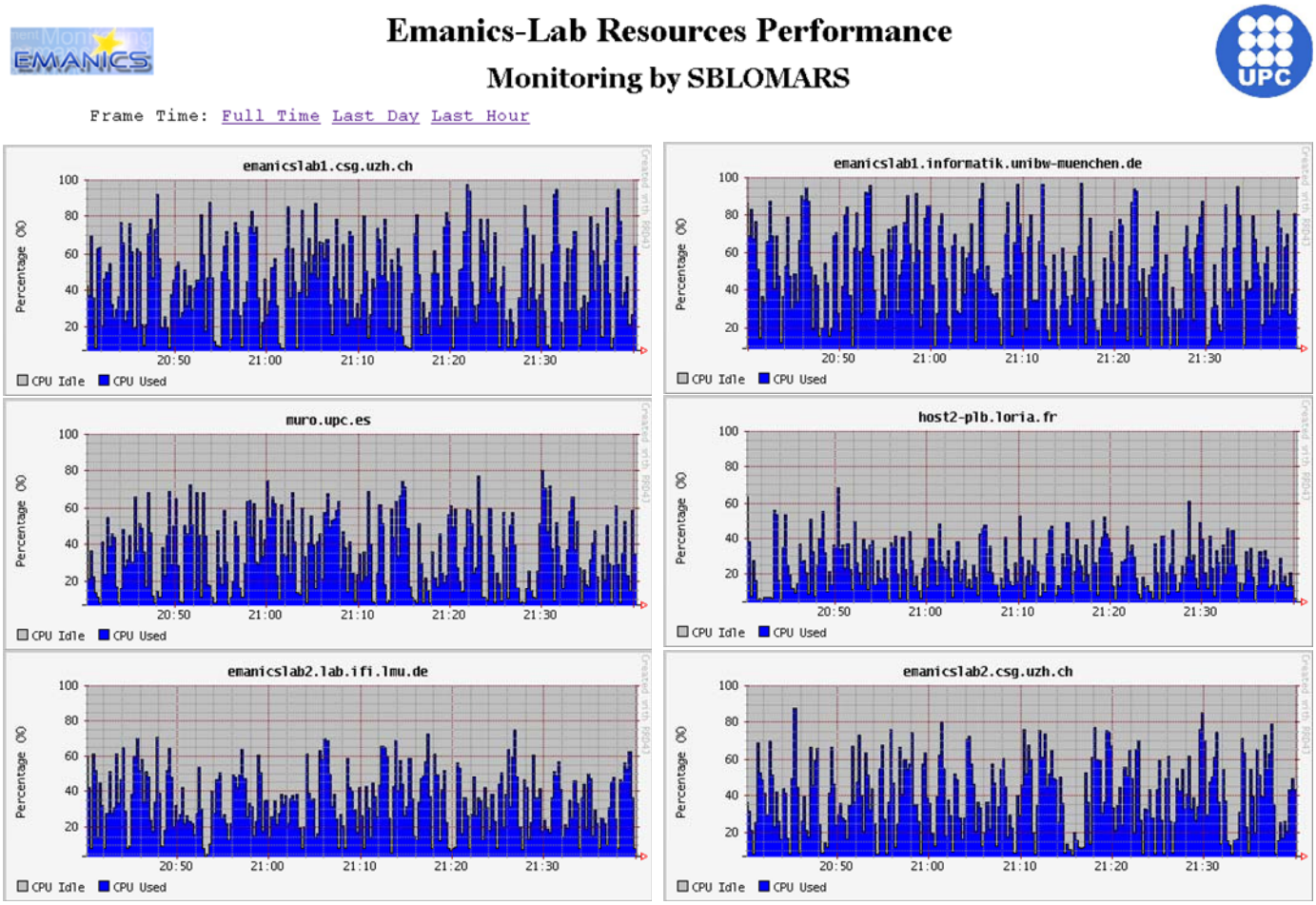

Figure 7. Resource utilization behavior of six EmanicsLab Nodes making use of the resource availability information to pre-select the destination nodes.

This behavior is very advisable in virtual environments. In fact, whereas in the first experiment when CPU is at 100% of its capacity the new incoming process will be queued until the CPU has completed previous jobs, in the second experiment it is not necessary to queue processes because nodes have idle CPU and then they have availability to execute more processes.

The adaptation of SBLOMARS to virtual environments has not been straightforward. We faced several problems especially with the use of SNMP that were finally solved. In its current version, SBLOMARS allow us to get data relative to the node where it is hosted. Our future work is focused in getting this protocol more integrated in the virtualization process with the aim to be able to obtain data at application level (slice level in EmanicsLab) and at virtual machine level.

ACKNOWLEDGMENT

This work has been partially done in the frameworks of the IST-EMANICS Network of Excellence (#26854). We specially acknowledge the contribution of Josep Tomás-Sanahuja.

REFERENCES

[1] European Network of Excellence for the Management of Internet Technologies and Complex Services, http://www.emanics.org/.

[2] L. Peterson, T. Anderson, D. Culler, and T. Roscoe: A Blueprint for Introducing Disruptive Technology into the Internet; First ACM Workshop on Hot Topics in Networking (HotNets), October 2002.

[3] A. Bavier, M. Bowman, B. Chun, et al.: Operating System Support for Planetary-Scale Services; First Symposium on Network Systems Design and Implementation (NSDI), March 2004.

[4] L. Peterson, A. Bavier, M. Fiuczynski, and S. Muir: Experiences Building PlanetLab; Seventh Symposium on Operating System Design and Implementation (OSDI), November 2006.

[5] Planetlab, http://www.planet-lab.org/.

[6] W. Stallings, "SNMP, SNMPv2, SNMPv3 and RMON 1 and 2, (Third Edition)". Addison-Wesley Professional, pages 365 - 398. 1999.

[7] M.L.Massie, B.N.Chun and D.E.Culler: The Ganglia Distributed Monitoring System: design, implementation and experience; Parallel Computing, Vol.30, Issue 7, July 2004.

[8] E. Magaña, L. Lefevre, and J. Serrat. "SBLOMARS: SNMP-based Balanced Load Monitoring Agents for Resource Scheduling in Large-scale Grids". Las Vegas, Nevada. USA 2007.

[9] Ganglia, http://ganglia.sourceforge.net/.

[10] Cisco Systems, Inc. "The Cisco IOS IP Service Level Agreements – White Paper". September, 2006.

[11] http://alcudia.upc.es:8080/emanics/

[12] UCD-SNMP: http://net-snmp.sourceforge.net/docs/mibs/ucdavis.html.

[13] SNMP Informant Windows Agents: http://www.wtcs.org/informant/.